FAST PROTECTION OF H.264/AVC BY SELECTIVE ENCRYPTION OF CABAC FOR I & P FRAMES

Z. Shahid, M. Chaumont and W. Puech

LIRMM Laboratory, UMR 5506 CNRS, University of Montpellier II
161, rue Ada, 34392 MONTPELLIER CEDEX 05, FRANCE
zafar.shahid@lirmm.fr, marc.chaumont@lirmm.fr, william.puech@lirmm.fr

ABSTRACT

This paper presents a novel method for the protection of copyrighted multimedia content. The problem of selective encryption (SE) is being addressed along with the compression for the state of the art video codec H.264/AVC. SE is performed in the context-based adaptive binary arithmetic coding (CABAC) module of video codec. For this purpose, CABAC is converted to an encryption cipher. It has been achieved through scrambling of equal length *bin strings*. In our scheme, CABAC engine serves the purpose of encryption cipher without affecting the coding efficiency of H.264/AVC by keeping exactly the same bit rate, generating completely compliant bitstream and requires negligible computational power. Owing to no escalation in bit rate, our encryption algorithm is better suited for real-time multimedia streaming. it is perfect for playback on handheld devices because of negligible increase in processing power. Nine different benchmark video sequences containing different combinations of motion, texture and objects are used for experimental evaluation of the proposed algorithm.

1. INTRODUCTION

With the rapid evolution of digital media, growth of processing power and availability of network bandwidth, several multimedia applications have emerged in the recent past. As digital data can be easily copied and modified, concerns regarding its protection and authentication have surfaced. Data encryption is used to restrict access of digital data to only authenticated users. For huge video data with real time constraints, SE is used in which only a small part of the whole bitstream is encrypted [13]. In this work, we have transformed CABAC module of H.264/AVC into encryption cipher. We have achieved this by scrambling of part of Exp-Golomb suffix of non-zero coefficients (NZs) and sign bits of all NZs.

SE of H.264/AVC has been studied in [6] wherein encryption has been carried out in some fields like intra-prediction mode, residual data, inter-prediction mode and motion vectors. Encryption for H.264/AVC has been discussed in [2] in which they do permutations of the pixels of macro-blocks (MBs) which are in *region of interest* (ROI). The drawback of this scheme is that bit rate increases as the size of ROI increases. This is due to change in the statistics of ROI as it is no more a slow varying region which is the basic assumption for video signals. The use of general entropy coder as encryption cipher has been studied in the literature in [15]. It encrypts NZs by using different Huffman tables for each input symbols. The tables, as well as the order in which they are used, are kept secret. This technique is vulnerable to known plaintext attacks as explained in [4]. Key-based interval splitting of arithmetic coding (KSAC) has used an approach [5] wherein intervals are partitioned in each iteration of arithmetic coding. Secret key is used to decide how the interval will be partitioned. Number of sub intervals in which an interval is divided should be kept small as it increases the bit rate of bitstream. Randomized arithmetic coding [3] is aimed at arithmetic coding but instead of partitioning of intervals like in KSAC, secret key is used to scramble the order of intervals. Encrypted bitstream (EB) compliance is a required feature for multimedia applications and both of these techniques make the bitstream non-compliant and hence, can not be decoded by standard H.264/AVC decoder. We have already presented a SE scheme of H.264/AVC based on context-based adaptive variable length coding (CAVLC) which fulfill real-time constraints by keeping the same bit rate and by generating completely compliant bitstream [12].

The rest of the paper is organized as follows. In Section 2, overview of H.264/AVC and CABAC is presented. It explains the working of CABAC along with its limitations from encryption point of view. We explain the whole system architecture in Section 3. Section 4 contains its experimental evaluation and performance analysis including its analysis over the wide range of QP values and its efficiency for different benchmark video sequences. In Section 5, we present the concluding remarks about the proposed scheme.

2. PRELIMINARIES

2.1 Overview of H.264/AVC

H.264/AVC [1] is the state of the art video coding standard of ITU-T and ISO/IEC. It offers better compression as compared to previous video standards. Like previous video standards, an input video frame is processed into blocks of 16x16 pixels, called macroblock (MB) and each of them is encoded separately. Each MB can be encoded as *intra* or *inter*. In *intra* frame, current MB is predicted spatially from MBs which have been previously encoded, decoded and reconstructed (neighboring MBs at top and left). In *inter* mode, motion compensated prediction is done from previous frames. The difference between original and predicted frame is call a *residual*. This *residual* is coded using transform coding followed by quantization and zigzag scan. In the last step, either of the entropy coding techniques namely CAVLC or CABAC is used. On the decoding side, compressed bitstream is decoded by entropy decoding module, followed by inverse-zigzag scan. These coefficients are then rescaled and inverse transformed to get the *residual* signal which is added to the predicted signal to reconstruct the original signal back.

H.264/AVC has some additional features as compared to previous video standards. Standard DCT transform has been replaced by integer transform (IT) [7] which does not need any multiplication operation and can be implemented by only additions and shifts. It can be implemented on 16 bit integer arithmetic and hence, has removed the problem of mismatch among codec implementations for different processor architectures. In *baseline* profile of H.264/AVC, It has 4x4 transform in contrast to 8x8 transform of previous standards. In higher profiles, it offers transform coding of adaptive size. It uses a uniform scalar quantization. For *inter* frame, H.264/AVC supports variable block size motion estimation namely 16x16, 16x8, 8x16 and 8x8 and the latter 8x8 block can be further divided up to 4x4 block size. Quarter pixel motion estimation, multiple reference frames, improved skipped and direct motion inference have also been included in this video codec. For *intra* frame, prediction has been shifted to spatial domain. Owing to all these additional features, H.264/AVC outperforms previous video coding standards [14].

2.2 Context-based Adaptive Binary Arithmetic Coding

In entropy coding, quantized transformed coefficients are scanned in reverse order (from high frequency to low frequency) as shown in Fig. 1. In this paper, we are presenting an encryption scheme based on CABAC [8]. CABAC is designed to better exploit the characteristics of NZs, consumes more processing and offers about 10% better compression than CAVLC on average [9]. Run-length coding has been replaced by *significant map* (SM) coding which specifies the position of NZs in the 4x4 block. Binary arithmetic coding module (BAC) of CABAC uses many context models to encode NZs and context model for a specific NZ depends on the number of NZs which have been already coded in the current block.

EB compliance is a required feature for some direct operations (displaying, time seeking, cutting, etc.). In each MB, header information is encoded first, which is followed by the encoding of MB data. EB is compliant if the following three conditions are fulfilled:

- To keep the bit rate of EB same as the original bitstream, encrypted *bin string* must have the same size as the original *bin string*.
- The encrypted *bin string* must be a valid codeword so that it may be decoded by entropy decoder.
- The decoded value of syntax element from encrypted *bin string* must fall in the valid range for that syntax element. Any syntax element which is predicted from neighboring MBs should not be encrypted. Otherwise the drift in the value of syntax element will keep on increasing and after few iterations, value of syntax element will fall outside the valid range and bitstream will be no more decodable.

To keep the bitstream compliant, we cannot encrypt MB header data, since it is used for prediction of future MBs. MB data contains NZs and **can be encrypted**. A MB is further divided into 16 blocks of 4x4 pixels to be processed by IT module. The *coded block pattern* (CBP) is a syntax element used to indicate which 8x8 blocks within a macroblock contain NZs. The *macroblock mode* (MBmode) is used to indicate whether a MB is *skipped* or not. If MB is not *skipped*, then MBmode indicates the

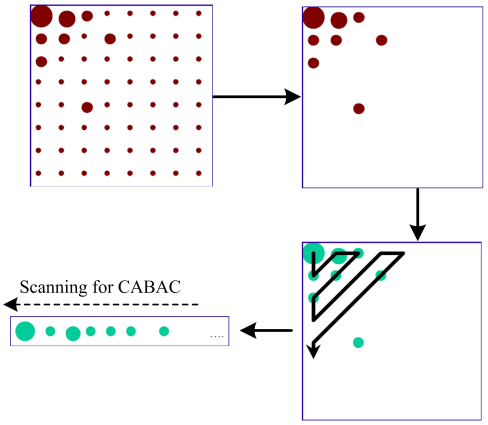

Figure 1: Scanning order of NZs in CABAC.

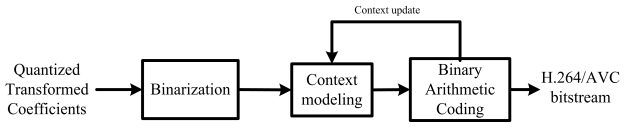

Figure 2: Block diagram of CABAC of H.264/AVC.

prediction method for a specific MB. For a 4x4 block inside MB, if CBP and MBmode are set, it indicates that this block is encoded.Inside 4x4 block, the *coded block flag* (CBF) is the syntax element used to indicate whether it contains NZs or not. CBF is encoded first. If CBF is zero, no further data is transmitted; otherwise, it is followed by encoding of SM. Finally, the absolute value of each NZ and its sign are encoded. Similar to MB header, header information of 4x4 block which includes CBF and SM, should not be encrypted for the sake of bitstream compliance.

CABAC consists of multiple stages as shown in Fig. 2. First of all, *binarization* is done in which, non-binary syntax element is converted to binary codeword called *bin string* which are more amenable to compression by BAC. Binary representation for a non-binary syntax element is done in such a way that it is close to minimum redundancy code. In CABAC, there are four basic code trees for *binarization* step, namely the *unary code*, the *truncated unary* code (TU), the *kth order Exp-Golomb* code (EGk) and the *fixed length* code (FL) as shown in Fig. 3.

For an unsigned integer value $x \geq 0$, the *unary code* consists of x 1's plus a terminating 0 bit . TU is only defined for x with $0 \leq x \leq s$. For $x < s$ the code is given by the unary code, whereas for $x = s$ the terminating 0 bit is neglected. EGk is constructed by a concatenation of a prefix and a suffix codeword and is suitable for binarization of syntax elements that represent prediction residuals. For a given unsigned integer value $x > 0$, the prefix part of the EGk codeword consists of a unary code corresponding to the length $l(x) = \left[log_2(\frac{x}{2^k} + 1)\right]$. The EGk suffix part is computed as the binary representation of $x + 2^k(1 - 2^{l(x)})$ using $k + l(x)$ significant bits. Consequently for EGk binarization, the number of symbols have the code length of $2l(x) + k + 1$. When k = 0, $2l(x) + k + 1 = 2l(x) + 1$.

FL is applied to syntax elements with a nearly uniform

distribution or to syntax elements, for which each bit in the FL *bin string* represents a specific coding decision e.g., CBF.

Three syntax elements are binarized by concatenation of these four trees, namely CBP, NZ and the motion vector difference (MVD). Binarization of absolute level of NZs is done by concatenation of TU and EG0 (UEG0). TU constitutes the prefix part with cutoff value $S = 14$. Binarization and subsequent arithmetic coding process is applied to the syntax element $coeff\_abs\_value\_minus1 = abs\_level - 1$, since quantized transformed coefficients with zero magnitude are encoded using SM.

For MVD, *bin string* is constructed by concatenation of TU and EG3 (UEG3). TU constitutes the prefix part with cutoff value $S = 9$. Suffix part of MVDs contains EG3 of $|MVD| - 9$ for $|MVD| > 9$ and sign bit.

Among all the four *binarization* techniques, The unary and TU codewords have different codeword length for each input value. They do not fulfill the first condition and their scrambling will change the bit rate of bitstream. Bit rate is very important factor for multimedia streaming applications over Internet and increase in bit rate affects the performance of such applications. Suffix of EGk and FL can be scrambled while keeping the bit rate unchanged. EGk is used for binarization of absolute value of levels and MVDs. Number of MVD *bin strings* have the same length. So it fulfills second condition and can be scrambled. But owing to the fact that MVDs are part of MB header and are used for prediction of future motion vectors, their encryption does not fulfill third condition and their encryption makes the bitstream non-compliant. The syntax elements which fulfill the criteria for encryption of H.264/AVC compliant bitstream are suffix of EG0 and sign bits of levels. Hence for each NZ with $|NZ| > 14$, encryption is done by scrambling of $l(x)$ bits to encrypt the EG0. It is followed by encryption of syntax element *coeff_sign_flag* which represents sign of levels of all non-zero levels. FL is used for binarization of syntax elements which belong to MB header and cannot be encrypted.

3. SYSTEM ARCHITECTURE

To keep the bit rate intact, we scramble the NZ with only those NZs whose EG0 *bin strings* have the same length. We initialize pseudo-random number generator (PRNG) with a secret key. EG0 codes, having same code length, constitute the scrambling space and it is dependent on the absolute value of NZ. The permutation space is $log_2(n+1)$ where n is the suffix part of absolute value of NZ. The block diagram of our scheme is shown in Fig. 4.

3.1 Encryption Process

The encryption process is shown in block diagram in Fig. 5. Let x be a suffix part of absolute level of NZ which is encoded using EG0 and is to be encrypted with the encrypted coefficients y can be given by:

$$y = (x + \gamma)\ mod\ log_2(x+1), \tag{1}$$

where γ is given by:

$$\gamma = rand()\ mod\ log_2(x+1). \tag{2}$$

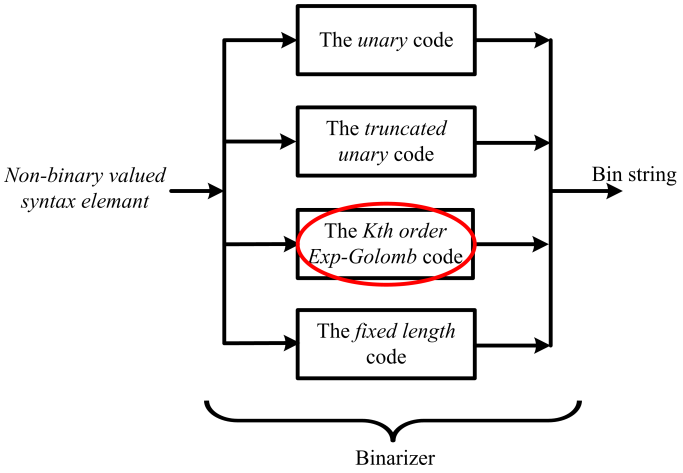

Figure 3: Block diagram of binarization stage illustrating different binarizatin methods.

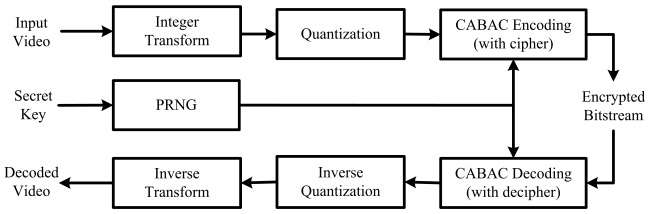

Figure 4: Block diagram of encryption and decryption process in H.264/AVC.

3.2 Decryption Process

For the decryption of NZ in H.264/AVC decoder, the process can be performed in reverse order in context-based binary arithmetic decoder (CABAD) module of H.264 decoder. Same secret key will be used as seed for PRNG to produce γ. Original value of EG0 suffix of $|NZ|$ can thus be extracted using encrypted NZ by using the formula:

$$x = (y + log_2(x+1) - \gamma)\ mod\ log_2(x+1). \tag{3}$$

4. EXPERIMENTAL RESULTS

We have used the reference implementation of H.264 JSVM 10.2 in AVC mode for video sequences in QCIF resolution. For the experimental results, nine benchmark video sequences have been used for the analysis. Each of them represents different combinations of motion (fast/slow, pan/zoom/rotation), color (bright/dull), contrast (high/low) and objects (vehicle, buildings, people). The video sequences 'bus', 'city' and 'foreman' contain camera motion while 'football' and 'soccer' contain camera panning and zooming along with object motion and texture in background. The video sequences 'harbour' and 'ice' contain high luminance images with smooth motion. 'Mobile' sequence contains a complex still background and foreground motion.

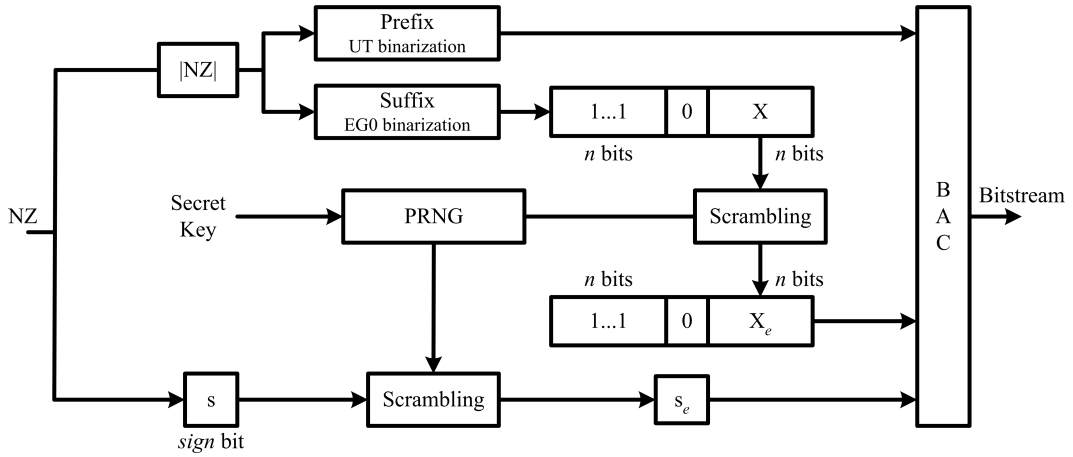

Figure 5: Encryption process for NZs and their signs in CABAC of H.264/AVC.

4.1 *Intra* Frames

To demonstrate the efficiency of our proposed scheme, we have compressed 100 frames of each sequence at 30 fps as *intra*. Fig. 6 shows the encrypted video frames at different *quantization parameter* (QP) values for the *foreman* video sequence. Their PSNR values are given in Table 1 and they are compared with the PSNR obtained for the same video frames without encryption. One can note that, whatever is the QP value, the quality of the encrypted video remains in the same lower range (below 10 dB on average for *luma*). It verifies that our algorithm is independent of QP value.
Table 2 compares the average PSNR of 100 frames of all benchmark video sequences at QP value 18 without and with SE. It confirms that this algorithm works well for various combinations of motion, texture and objects and is significantly efficient. Although it depends on the contents of video and the quantization value, yet in proportion to overall computation which a video codec consumes, it is negligible. Average PSNR value of *luma* for all the sequences at QP value 18 is 9.73 dB.

4.2 *Intra & Inter* Frames

Video data normally consists of an *intra* and a trail of *inter* frames. *Intra* frames are inserted periodically to restrict the drift because of lossy compression and rounding errors. In this experimental evaluation, *intra* period is set at 10 in a sequence of 100 frames. Results shown in table 3 verifies the effectiveness of our scheme over the whole range of QP values for *foreman*. Table 4 verifies the performance of our algorithm for all video sequences for *Intra & Inter* frames. Average PSNR of *luma* for all the sequences is 9.68 dB.

5. CONCLUSION

In this paper, a novel framework for SE of H.264/AVC based on CABAC has been presented. Real-time constraints have been handled successfully by having the same bit rate and by having compliant bitstream. The encrypted bitstream is H.264/AVC format compliant and can be played back by any standard H.264/AVC decoder. Owing to no escalation in bit rate, our encryption scheme is suitable for heterogeneous multimedia streaming scenarios in real-time environment.

Table 1: Comparison of PSNR without encryption and with SE for *foreman* sequence at different QP values for *intra* frames.

QP	PSNR (Y) (dB)		PSNR (U) (dB)		PSNR (V) (dB)	
	Without SE	With SE	Without SE	With SE	Without SE	With SE
12	50.05	8.92	49.99	24.08	50.78	23.84
18	44.43	8.42	45.62	23.87	47.42	22.14
24	39.40	8.38	41.70	24.87	43.86	22.70
30	34.93	8.92	39.38	24.60	40.99	22.71
36	30.80	8.89	37.33	24.65	38.10	22.90
42	27.03	8.93	35.87	24.24	36.41	23.94

Table 2: Comparison of PSNR without encryption and with SE of benchmark video sequences at QP 18 for *intra* frames.

Seq.	PSNR (Y) (dB)		PSNR (U) (dB)		PSNR (V) (dB)	
	Orig.	SE	Orig.	SE	Orig.	SE
bus	44.26	7.73	45.22	25.19	46.50	26.86
city	44.28	11.52	45.83	30.50	46.76	31.86
crew	44.81	9.39	45.81	23.80	45.66	19.90
football	44.59	11.46	45.70	15.79	45.98	23.10
foreman	44.43	8.42	45.62	23.87	47.42	22.14
harbour	44.10	9.48	45.60	23.82	46.63	31.20
ice	46.56	10.37	48.70	25.42	49.19	19.73
mobile	44.45	8.42	44.14	13.47	44.04	11.11
soccer	44.26	10.84	46.59	19.69	47.82	24.83

Table 3: Comparison of PSNR without encryption and with SE for *foreman* sequence at different QP values for *intra* and *inter* frames.

QP	PSNR (Y) (dB)		PSNR (U) (dB)		PSNR (V) (dB)	
	Without SE	With SE	Without SE	With SE	Without SE	With SE
12	49.54	8.41	49.89	23.34	50.63	22.16
18	43.91	9.23	45.50	26.06	47.55	21.11
24	38.90	8.61	42.04	24.62	44.29	21.83
30	34.59	9.19	39.84	24.02	41.56	25.18
36	30.76	8.78	37.96	25.12	38.86	23.50
42	26.61	8.31	36.34	25.30	36.92	27.06

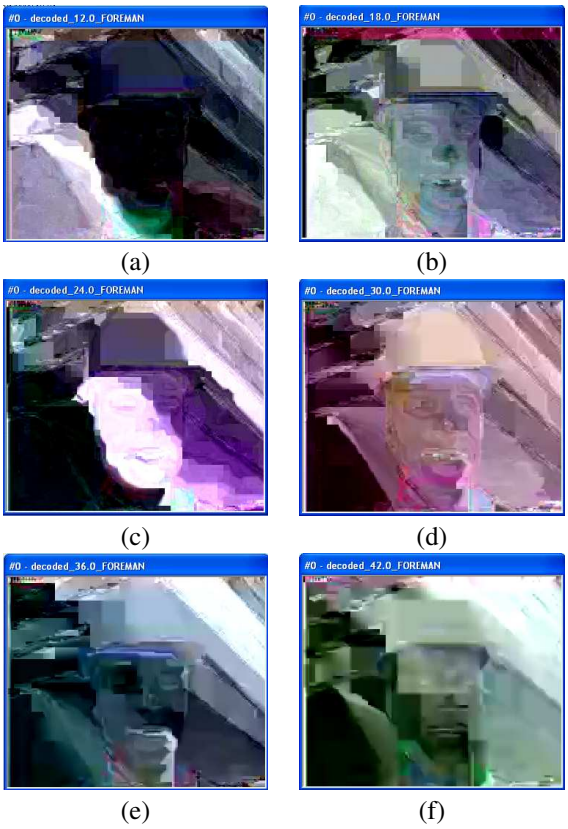

Figure 6: Decoding of encrypted video *"foreman"*: first frame with QP equal to (a) 12 (b) 18 (c) 24 (d) 30 (e) 36 (f) 42.

The experiments have shown that we can achieve the desired level of encryption in each frame, while maintaining the H.264/AVC format compliance for both *intra* and *inter*, under a minimal set of computational requirements. Hence, it is perfect for multimedia playback on handheld devices. The proposed system can be extended for ROI specific video protection [11] for video surveillance and can be applied to medical image transmission [10].

Acknowledgment

This work is in part supported by the VOODDO (2008-2011) project of the french ANR (Agence Nationale pour la Recherche).

REFERENCES

[1] Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 / ISO/IEC 14496-10 AVC). Technical report, Joint Video Team (JVT),Doc. JVT-G050, March 2003.

[2] P. Carrillo, H. Kalva, and S. Magliveras. Compression Independent Object Encryption for Ensuring Privacy in Video Surveillance. In *ICME*, 2008.

[3] M. Grangetto, E. Magli, and G. Olmo. Multimedia Selective Encryption by Means of Randomized Arithmetic Coding. *IEEE Transactions on Multimedia*, 8(5):905–917, Oct. 2006.

[4] G. Jakimoski and K. Subbalakshmi. Cryptanalysis of Some Multimedia Encryption Schemes. *IEEE Transactions on Multimedia*, 10(3):330–338, April 2008.

Table 4: Comparison of PSNR without encryption and with SE of benchmark video sequences at QP 18 for *intra* and *inter* frames.

Seq.	PSNR (Y) (dB)		PSNR (U) (dB)		PSNR (V) (dB)	
	Orig.	SE	Orig.	SE	Orig.	SE
bus	43.72	7.44	45.10	25.06	46.44	28.03
city	43.80	10.84	45.73	30.07	46.78	32.24
crew	44.45	8.83	45.81	23.00	45.71	20.34
football	44.15	11.52	45.71	12.65	46.05	23.50
foreman	43.91	9.23	45.50	26.06	47.55	21.11
harbour	43.70	9.71	45.44	26.05	46.57	32.52
ice	46.13	9.85	48.63	24.37	49.14	21.27
mobile	43.84	8.94	44.15	12.74	44.06	11.52
soccer	43.53	10.76	46.45	20.12	47.75	23.84

[5] W. Jiangtao, K. Hyungjin, and J. Villasenor. Binary arithmetic coding with key-based interval splitting. *IEEE Signal Processing Letters*, 13(2):69–72, Feb. 2006.

[6] S. L. S. Z. Liu, Z. Ren, and Z. Wang. Selective Video Encryption Based on Advanced Video Coding. *Lecture notes in Computer Science, Springer-verlag*, (3768):281–290, 2005.

[7] H. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky. Low-complexity transform and quantization in H.264/AVC. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):598–603, 2003.

[8] D. Marpe, H. Schwarz, and T. Wiegand. Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):620–636, July 2003.

[9] I. Moccagatta and K. Ratakonda. A performance comparison of CABAC and VCL-based entropy coders for SD and HD sequences. Oct. 2002.

[10] W. Puech and J. Rodrigues. A New Crypto-Watermarking Method for Medical Images Safe Transfer. In *Proc. 12th European Signal Processing Conference (EUSIPCO'04)*, pages 1481–1484, Vienna, Austria, 2004.

[11] J.-M. Rodrigues, W. Puech, and A. Bors. Selective Encryption of Human Skin in JPEG Images. In *Proc. IEEE Int. Conf. on Image Processing, Atlanta, USA*, pages 1981–1984, Oct. 2006.

[12] Z. Shahid, M. Chaumont, and . W. Puech . Fast protection of h.264/avc by selective encryption. In *SinFra 2009, Singaporean-French IPAL Symposium, Fusionopolis,*, Singapore, 18-20 feb. 2009.

[13] A. Uhl and A. Pommer. *Image and Video Encryption: From Digital Rights Management to Secured Personal Communication*. Springer, 2005.

[14] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, July 2003.

[15] C.-P. Wu and C.-C. Kuo. Design of Integrated Multimedia Compression and Encryption Systems. *IEEE Transactions on Multimedia*, 7:828–839, 2005.